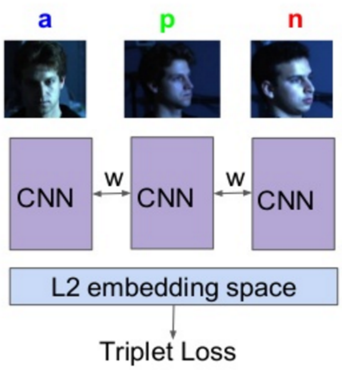
Triplet Loss

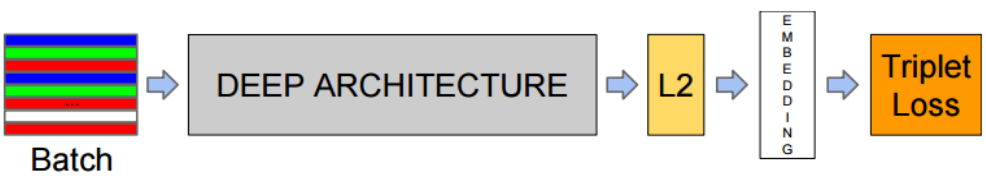
Introduction

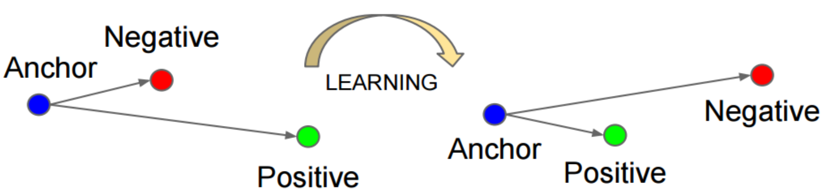
The triplet loss is ofen uses on perseon re-indentification or face recognition problems, the idea of triplet loss is to receive 3 scores, with 1 pair related to the score of similar classes and an extra score related to a "negative class". The training objective will minimize this loss by making related scores similar while making the difference between the related scores and the negative score larger.

$$Loss_{triplet} = \sum_{i=1}^{N} \left[L_{2_{distance}} \left(f^a - f^p \right)^2 - L_{2_{distance}} \left(f^a - f^n \right)^2 + \alpha \right]$$

The triplet loss has 3 inputs, which are the output of a neural network for 3 classes, a class called "anchor class", another class that is positive, and a negative class. This concept for person reindentification or face recognition is important because for example the anchor class could be a person, and positive class could be the same person under other circunstance, and a negative class.







Here the inputs of the triplet loss comes from 3 different networks what share the same weights, but this also can be achieved by using a single network and building the batch with the anchor, positive and negative class. This also implies that if the triplet loss has 3 inputs during backpropagation we should return 3 gradients.

Euclidian distance (Or L2 distance)

Like other distances (ie L1 distance) the L2 distance return a scalar that represent how 2 vectors(p,q) are similar to each other.

$p := [1, 2, 3] = [1, 2, 3]$

$q := [1.1, 2, 3.3] = [1.1, 2, 3.3]$

$$L_2 := (a, b, N) \rightarrow \sqrt{\sum_{i=1}^{N} (a[i] - b[i])^2}$$

$$(a, b, N) \rightarrow \sqrt{\sum_{i=1}^{N} (a_i - b_i)^2} \quad \textbf{(1)}$$

$L_2(p, q, 3) = 0.3162277660$

$$L_2(p_v, q_v, 3) = \sqrt{(p_{v_1} - q_{v_1})^2 + (p_{v_2} - q_{v_2})^2 + (p_{v_3} - q_{v_3})^2}$$

On the context of loss functions we use the square l2 distance to actually cancel the squar-root term.

$$\| p_i - q_i \|_2^2 = \left(\sqrt{\sum_{i=1}^{N} (a[i] - b[i])^2} \right)^2$$

This sometimes causes confusion because now it will be called L2-norm loss

$$L_{2_{loss}} := (a, b, N) \rightarrow \sum_{i=1}^{N} (a[i] - b[i])^2$$

$$(a, b, N) \rightarrow \sum_{i=1}^{N} (a_i - b_i)^2 \quad \textbf{(2)}$$

$L_{2_{loss}}(p, q, 3) = 0.10$

$$L_{2_{loss}}(p_v, q_v, 3) = (p_{v_1} - q_{v_1})^2 + (p_{v_2} - q_{v_2})^2 + (p_{v_3} - q_{v_3})^2$$

There is a variant of the L2-norm loss called MSE (Mean squared error) loss which divide the L2-norm loss by the batch-size, the idea is to decouple the batch size from the loss value.

$$MSE_{loss} := (a, b, N) \rightarrow \frac{1}{N} \cdot \left(\sum_{i=1}^{N} (a[i] - b[i])^2 \right) = (a, b, N) \rightarrow \frac{\sum_{i=1}^{N} (a_i - b_i)^2}{N}$$

$$MSE_{loss}(p_v, q_v, 3) = \frac{1}{3} (p_{v_1} - q_{v_1})^2 + \frac{1}{3} (p_{v_2} - q_{v_2})^2 + \frac{1}{3} (p_{v_3} - q_{v_3})^2$$

Get the derivative of L2-norm loss w.r.t to input

Before we delve into the gradient of the triplet-loss let's investigate how to find the derivative of the L2-norm function, by definition the idea of the norm function is to give a positive scalar number that represent the size of a vector, so we can consider the following simplification.

$\sum_{i=1}^{N} (a[i] - b[i])^2 \Rightarrow \sum_{i=1}^{N} u^2$, where u will be a function that calculate the difference $(a[i] - b[i])$ vector

Let's just lay out some derivatives rules, consider u as a function

Scalar multiple rule: $\frac{d}{dx}(\alpha \cdot u) = \alpha \frac{d}{dx} u$

Sum Rule: $\frac{d}{dx} \sum u = \sum \frac{d}{dx} u$, this is nice because I don't know yet how to calculate derivatives with sum properly on maple

$$sum \cdot \frac{\partial}{\partial a} (a - b)^2 = sum\ (2\,a - 2\,b)$$

$$sum \cdot \frac{\partial}{\partial b} (a - b)^2 = sum\ (-2\,a + 2\,b)$$

Putting all together

Now let's put together the L2-norm inside the triplet loss, and also push outside all the $\sum$ operators

$$f := (anchor, positive, negative, \alpha) \rightarrow [(anchor - positive)^2 - (anchor - negative)^2 + \alpha]$$

$$(anchor, positive, negative, \alpha) \rightarrow [(anchor - positive)^2 - (anchor - negative)^2 + \alpha] \tag{3}$$

$$\frac{d}{d\,anchor} f(anchor, positive, negative, \alpha) = [-2\,positive + 2\,negative]$$

$$\frac{d}{d\,positive} f(anchor, positive, negative, \alpha) = [-2\,anchor + 2\,positive]$$

$$\frac{d}{d\,negative} f(anchor, positive, negative, \alpha) = [2\,anchor - 2\,negative]$$

$$factors(-2\,positive + 2\,negative) = [2, [[negative - positive, 1]]]$$

$$factors(-2\,anchor + 2\,positive) = [-2, [[anchor - positive, 1]]]$$

$$factors(2\,anchor - 2\,negative) = [2, [[anchor - negative, 1]]]$$

References

- https://en.wikipedia.org/wiki/Loss_function
- http://stackoverflow.com/questions/33330779/whats-the-triplet-loss-back-propagation-gradient-formula
- http://www.cnblogs.com/wangxiaocvpr/p/5452367.html
- https://en.wikipedia.org/wiki/Euclidean_distance
- http://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-image-retrieval-upc-2016
- https://www.physicsforums.com/threads/derivative-of-a-function-involving-square-root-of-sum-of-squares.643123/
- http://math.stackexchange.com/questions/84331/does-this-derivation-on-differentiating-the-euclidean-norm-make-sense
- http://math.stackexchange.com/questions/883016/gradient-of-l2-norm-squared
- https://davidrosenberg.github.io/ml2015/docs/3a.loss-functions.pdf
- https://davidrosenberg.github.io/ml2015/#home
- https://devblogs.nvidia.com/parallelforall/understanding-aesthetics-deep-learning/
- http://rishy.github.io/ml/2015/07/28/l1-vs-l2-loss/
- http://nbviewer.jupyter.org/github/rishy/rishy.github.io/blob/master/ipy_notebooks/L1%20vs.%20L2%20Loss.ipynb
- http://mccormickml.com/2014/03/04/gradient-descent-derivation/
- https://www.quora.com/What-is-the-difference-between-L1-and-L2-regularization
- http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/